Turn-Based RPG Combat System – Data Modeling Project

Assignment 2 – Database Systems

Name: Tomáš Meravý Murárik
Student ID:127232
Submission Date: 13 4 2025

Table of Contents

Functional Specification

World: This game is a 2D turn-based game (with no fixed turn order) in which characters fight against each other and explore the world. Due to the task specifications, I will only describe the combat and healing aspects of my game, omitting movement and world interaction. Characters may enter combat and fight each other. The main tables in my program are character, action, class, and action_type, which form the core of my database

Entities:

character:

Characters represent the in-game character of the player.
Attributes:

- id INT [primary]UNIQUE
- name VARCHAR(255)
- class_id INT [foreign]
    - class_id is connected to class tables
- health INT
- action_points INT
- strength INT
- dexterity INT
- constitution INT
- intelligence INT
    - (attributes from health to intelligence represent the current value of each stat)
- in_combat BOOL
    - (helper variable indicating if the character is in combat)

class:

This entity provides players with different ways to play the game by offering class-specific advantages and combat bonuses.
Attributes:

- id INT [primary]UNIQUE
- name VARCHAR(255)
- health INT
- action_points INT
- strength INT
- dexterity INT
- constitution INT
- intelligence INT
- armorClass INT
- inventorySize INT
    - (attributes from health to inventorySize are class bonuses)

action:

Action table stores all actions performed throughout the game's history, including movement, spells, and melee attacks.
Attributes:

- id INT [primary]UNIQUE

- action_type_id INT [foreign]
  - action_type_id is connected to action_type.
- target INT [foreign]
  - target is connected character
- character_id INT [foreign]
  - character_id is connected to character .
- combat_id INT [foreign] NULL
  - combat_id is connected to combat. (It can be NULL . It’s NULL when the character is out of combat.)
- item_id INT [foreign] NULL
  - (references the item table; indicates whether an item was used during this action and displays the item’s id if so.)

- effect_value INT
  - (displays how much damage , healing will be done or stun duration.)
- hit BOOL
  - (Informs if the action hit or missed.)

action_type:

Action Type is a blueprint for all actions doable in this game whether it’s in or outside of combat. Anything player can do is defined as action in action type table.
Attributes:

- id INT [primary] UNIQUE
- name VARCHAR(256)
- action_category_id INT [foreign]
  - connects this table to action_category table
- item_id INT [foreign] NULL
  - connecsts this table to items table. (This column is here in case the action requires a specific item.)
- description VARCHAR(8192)
- effect ENUM(“stun”,”demage”,”heal”)
  - (stund_duration , damage and heal are all values relevant to spells and attacks with value. One action may have only one effect.)
- cost INT
- effect_value INT

action_category:

This table stores all action categories (e.g., fire spell, water spell, item attack, out-of-combat action) along with their associated costs.
Attributes:

- Id INT [primary] UNIQUE
- name VARCHAR (256)
- cost INT

class_attribute_modifier:

This bridge table connects classes and action types. Each action (including spells) has its own modifier based on the character's class. When a spell is cast, the game first checks the character's class and then retrieves the corresponding modifiers from this table.
Attributes:

- action_category_id INT [foreign]
  - connects this table with action_category table
- class_id INT [foreign]
  - connects this table with class table
- attribute ENUM("strength","dexterity","constitution","intelligence")
- value INT
  - (this column dictates the value of the action modifier)

character_actions:

May also be called the spellbook, this bridge table records every action that each character can perform, including healing, moving, fireball, cleave, and more.
Attributes:

- character_id INT [foreign]
- action_type_id INT [foreign]

combat:

This table records the state of combat and each of its rounds, enabling relations with the playground.
Attributes**:**

- id INT [primary] UNIQUE
- combat_num INT
  - (Note: The id and combat_num are different; combat_num combined with round identifies the exact round of combat, while the id is used to connect these rows with other tables.)
- round INT
- status ENUM("on-going","finished")
  - (An ENUM is used instead of a boolean for clearer understanding)

item:

This table contains items that players may acquire by looting and use in combat.
Attributes:

- id INT [primary] UNIQUE
- name VARCHAR(256)
- description VARCHAR(8192)
- base_damage INT
- weight INT

playground:

This table is used to store which items were available on the ground at the start of specific combat rounds. It serves as a bridge between the combat and item tables.
Attributes:

- combat_id INT [foreign]
    - connects this table with combat table
- item_id INT [foreign]
    - connects this table with item table

item_attribute_modifier:

This bridge table connects items to their modifier attributes, which vary based on the class of the wielder.
Attributes:

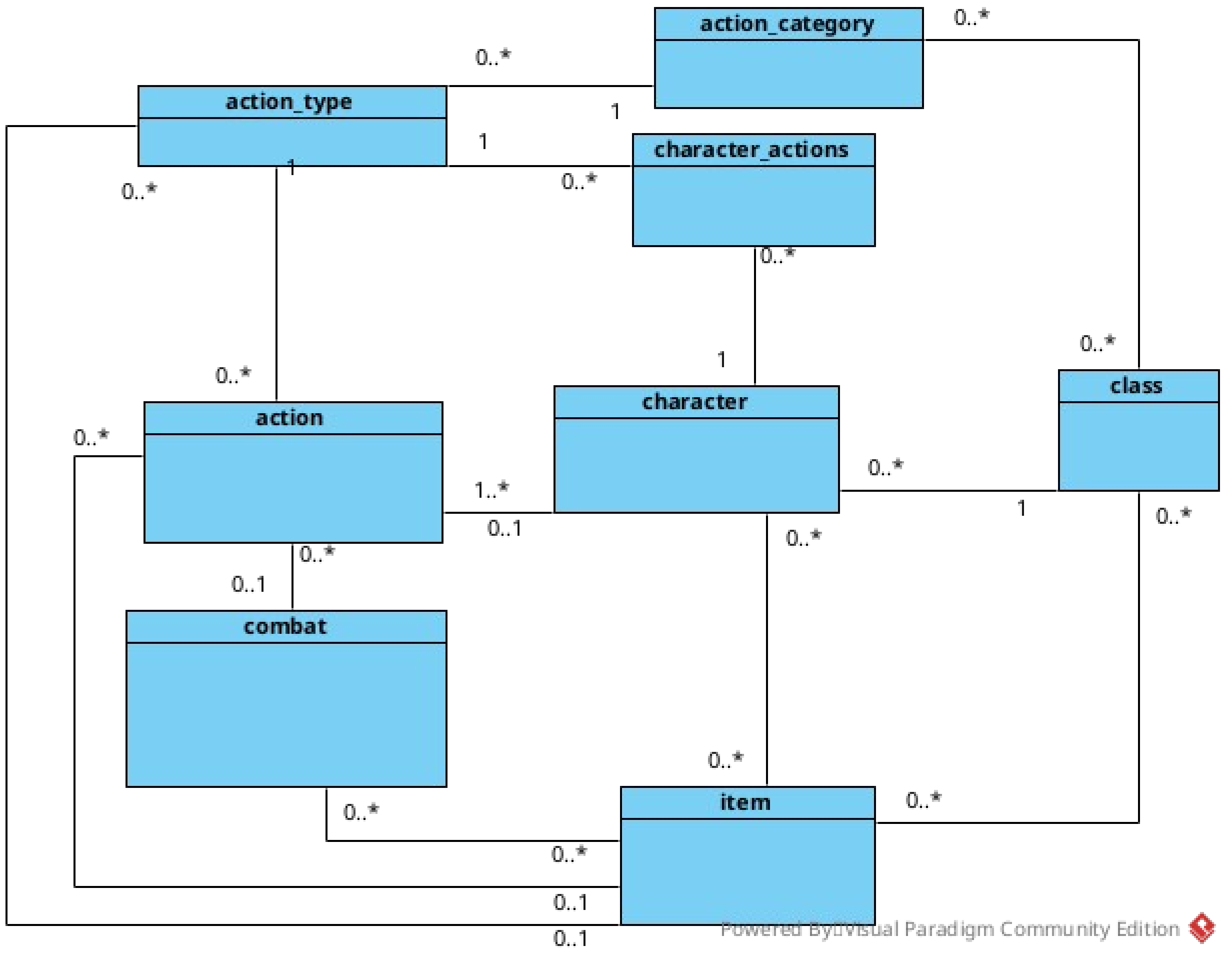
- id INT [primary] UNIQUE
- item_id INT [foreign]
    - connects this table with item table
- class_id INT [foreign]
    - connects this table with class table
- attribute ENUM(“strength”,”dexterity”,”constitution”,”intelligence”)
- value INT

inventory:

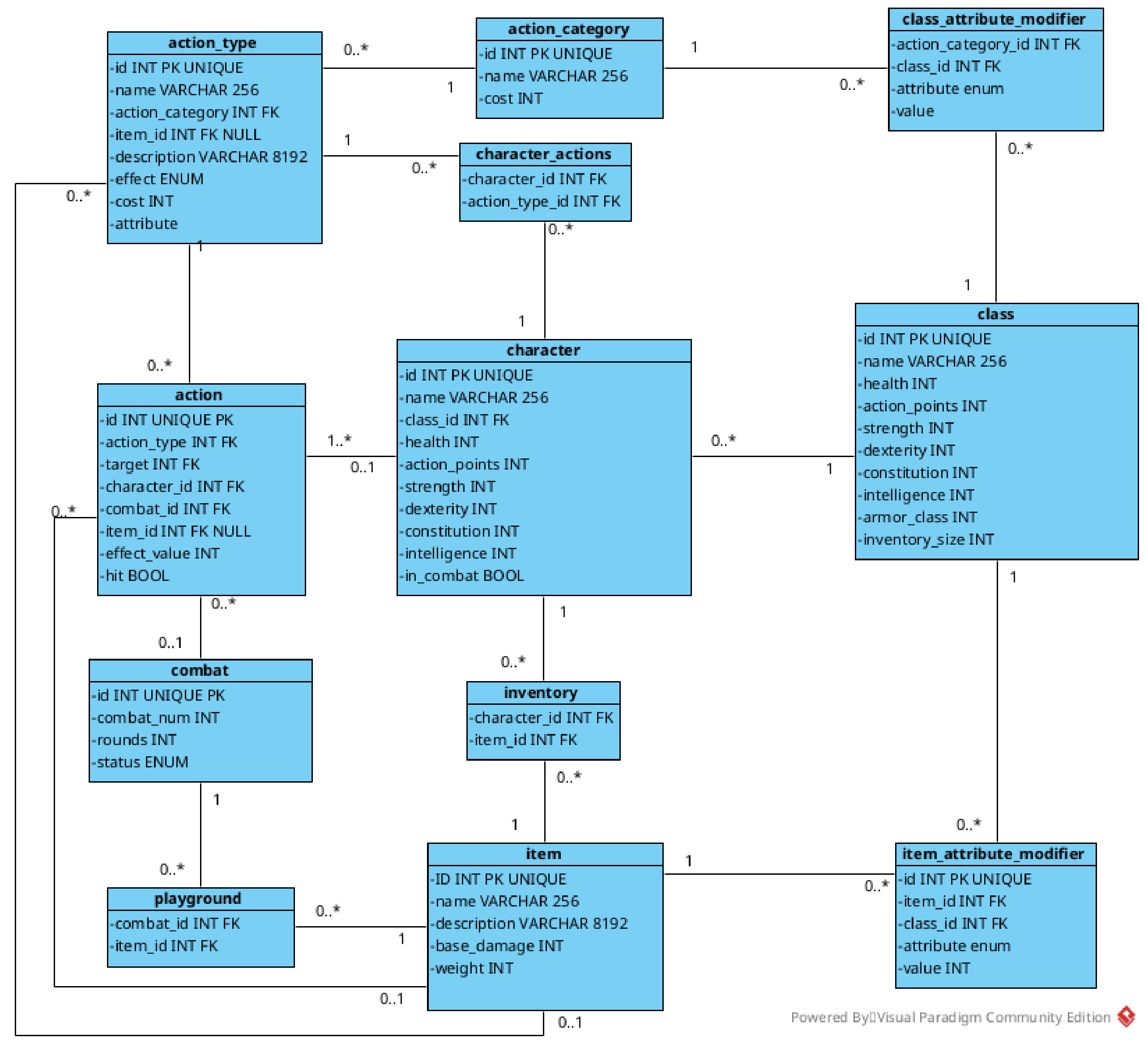
This bridge table connects characters to the items in their in-game inventory.
Attributes:

- character_id INT [foreign]
    - connects this table with character table
- item_id INT [foreign]
    - connects this table with item table

E-R model


action_category
action_type
character_actions
action
character
class
combat
item
0..*
1
0..1
1..*
Powered By Visual Paradigm Community Edition

Logical Model



Entity Relationship Descriptions

1. Character owns Inventory
2. Inventory contains Items
3. Character performs Actions
4. Action targets Character
5. Action uses Items
6. Action occurs in Combat
7. Action has an Action Type
8. Action Type belongs to an Action Category
9. Character has a Class
10. Class defines base attributes
11. Class modifies Action Categories via Class Attribute Modifiers
12. Item has Attribute Modifiers
13. Item can be used in an Action
14. Combat contains Actions
15. Combat occurs on a Playground
16. Playground contains Items

Key Processes

Before explaining key processes I'd like to define few rules of my game.

1. Player starts combat by performing an attack on another character who's within range of that attack (This could be defined by adding X and Y position to the character but since these kinds of details would be unnecessary for this task I decided to leave them out).
2. A player may enter combat at any stage of the game by walking into the combat ground (a 2D grid clearly defined on the map). A player may leave combat by exiting the combat ground, but may not re-enter combat with the same combat_num.
3. A player may perform any kind of action outside combat for 0 action points. This includes healing, moving, and dealing damage.
4. In combat, any character may target any other character, meaning that range provides no advantage.

Define key processes:

Key process 1 and 4)

What happens when characters cast a spell inside a combat? Which entities / records are created? What attributes are changing? What have to be checked?
How does the system handle a spell being cast (with all calculations and hit chances)?

When a character casts a spell in combat, the game first checks whether the character owns the spell by verifying their character_id in the character_actions table. If the player owns the spell and has any required items for the action, the game calculates the cost of the spell using the formula:

EffectiveCost = baseCostCategory.modifier – (1-(selectedAttribute/100))(1-itemModifier)

which in my tables would be:

EffectiveCost = action_category[cost]- (1-(class_attribute_modifier[value]/100))(1-item_attribute_modifier[value])

If the character doesn't have enough action_points, game will forbid this action. In case the character does have enough action_points, the game will proceed to look if the target character is in this combat (this can be done on server side or also by doing a query and checking if the target character has any actions in with this combat_num while his last action isn't "left_combat"). Upon confirming that the target character is in combat, the game calculates the damage as follows:

Damage = BaseDamage(1 + ConfiguredAttribute/20)

which in my tables will be

action[effect_value]= action_type[effect_value](1+class_attribute_modifier[value]/20).

Next, the game simulates a d20 roll, adds the class_attribute_modifier[value] to the roll, and compares the result with the target's Armor Class, calculated as:

ArmorClass = 10 + (character[dexterity]/2) + class[armorClass]

If this action results in hit then firstly the program will proceed to adding these values into the table as so:

1. Add entry to the action table.
2. Based on effect value of action type perform one of the following actions :
   1. subtract action[effect_value] from health of targer.
   2. Forbid target from moving for action[effect_value] number of rounds
   3. Increment target's health by action[effect_value]
3. Decrease action_points of character who casted the spell by EffectiveCost.

If the action does not hit, the software will follow these steps:

1. Add entry into the action table
2. subtract EffectiveCost from characters action_points

Note that if the character casts an action on themselves, it always counts as a hit and the d20 roll is skipped.

In conclusion:
- character table gives us necessary attributes.
- class tables provides modifiers for both caster and target character.
- action table is the real entry of the log.
- action_type provides the blueprints of the action used.
- action_category provides us with cost attribute needed in our equation.
- character_actions is used to check whether the casting character owns the spell
- class_attribute_modifier is collection of all modifiers on action_category of the action being used.

Key process 2)

How does a character rest and recover health?
Characters can rest and recover health both in and outside of combat. If the character is outside of combat, they can simply use a rest action, which instantly restores their health to the maximum HP as calculated by:

MaxHp=10+character[constitution](class[constitution]/2)

This action first checks that the character’s in_combat flag is false. On the other hand, if the character is in combat and belongs to a class that includes healing magic (for example, a druid), they will recover an amount of HP as described in Key Process 1, but cannot exceed MaxHp.
Both actions are logged in the actions table with both the character and target fields set to the ID of the character healing themselves and the effect_value reflecting the amount of HP restored, after which the character’s health is incremented.

Key process 3)

What happens when they enter combat?
A character enters combat by attacking another character. When this occurs, a square grid around them is designated as the combat ground. The server first adds a new entry to the combat table with a unique combat_num and sets the round to 1; it then creates item entries in the playground table. Next, the server logs an entry in the action table with the following values:
- action_type: “entered_combat”
- target:<character_id>
- character_id:<character_id of caster of the spell>
  - this value also serves as indicator of who started the combat.
- item_id:NULL
- combat_id:<newly created entry in combat table>
- effect_value:0
- hit:1

This action is logged for every character present on the combat ground (with the appropriate combat_id for each). Only after these steps does the server execute the attack on the target, assigning the new combat_id from the combat table to that action and to all subsequent actions in the combat.

Key process 4)

This process has been merged with Key Process 1. Please refer to Key Process 1 for the complete description.

Key process 5)

How are items added to a combat?
Items can appear in combat in two ways

1. At the start of the first round.
2. By dropping from a character upon death or by deliberate action.

For the first method, when combat begins (i.e., when one player hits another), the game rolls a d20. For every roll above 17, the game creates one random item and adds it to the playground table with a combat_id referencing the newly created combat entry. This new item can be held and used by any character because there are no restrictions on class-specific item usage (for example, a support character may wield a great sword if desired).

The second method is a bit more complex. When a character dies, the server keeps all items the player owned by adding them to a list of items already on the ground. The system waits until the round ends, adjusting the list based on character actions. When a new round starts, new entries in the playground are created with the combat_id referencing the new round and all items from the stored list are added to the playground table.
Note that items are not immediately added to the database when someone dies. They are only displayed as items available at the start of the round.

Key process 6)

How does a character loot an item and check inventory limits?

To pick up an item it needs to be on the combat ground. When a character is about to pick up an item the server first checks if it's in his list of all items on combat ground. After confirming that it is the server proceeds to calculate the maximum weight character can carry by the following equation:

MaxInventoryWeight = (Strength + Constitution)ClassInventoryModifier

which in my tables will look like:

MaxInventoryWeight = (character[strength] + character[constitution])class[inventorySize]

After getting the maximum weight a character can carry at the moment the system will sum up all weights of items in inventory with character_id of the character picking up the item. If the added weight of all carried items added to the new item does not exceed MaxInventoryWeight the software allow this action and add this item to inventory of the characetr. Otherwise the system will refuse this action and won't add anything into any table.